Submitted, 2017 Internationl Symposium on Robotics Research (ISRR)
http://www.cds.caltech.edu/~murray/preprints/dfm17-isrr_s.pdf

Parallelizing Synthesis from Temporal Logic Specifications by Identifying Equicontrollable States

Sumanth Dathathri, Ioannis Filippidis and Richard M. Murray

Abstract
Keywords: Formal Methods, Discrete Event Systems, High-Level Controller Synthesis, Verification

For the synthesis of correct-by-construction control policies from temporal logic specifications the scalability of the synthesis algorithms is often a bottleneck. In this paper, we parallelize synthesis from specifications in the GR(1) fragment of linear temporal logic by introducing a hierarchical procedure that allows decoupling of the fixpoint computations. The state space is partitioned into equicontrollable sets using solutions to parameterized reachability games that arise from decomposing the original GR(1) game into smaller reachability games. Following the partitioning, another synthesis problem is formulated for composing the strategies from the decomposed reachability games. The formulation guarantees that composing the synthesized controllers ensures satisfaction of the given GR(1) property. Benchmarking experiments with robot planning problems demonstrate good scalability of the approach.

1 Introduction

As robotic systems get more complex, logic specifications assist in precisely specifying desired behavior for a system and constructing controllers that provably guarantee satisfaction of the specification. In our work, we focus on reactive synthesis from temporal logic specifications. This involves reasoning about all admissible

Sumanth Dathathri
CMS, California Institute of Technology, USA, e-mail: sdathath@caltech.edu

Ioannis Filippidis
CDS, California Institute of Technology, USA, e-mail: ifilippi@caltech.edu

Richard M. Murray
CDS, California Institute of Technology, USA, e-mail: murray@cds.caltech.edu

behaviors for the environment and synthesizing a policy for the controlled agent. This makes the algorithms for synthesis difficult to scale and synthesis can be prohibitively expensive when applied to problems with large state spaces.

In this paper, we focus on reactive planning for specifications in linear temporal logic (LTL) and, in particular, the *Generalized Reactivity (1)* (GR(1)) fragment. The complexity for synthesis for general LTL specifications is doubly exponential in the length of the formula [16]. For LTL formulas in the GR(1) fragment, synthesis can be performed in time polynomial in the size of the state space, which is typically exponential in the number of variables that describe the problem. The complexity of synthesis for the GR(1) fragment scales as cubic or quadratic depending on the algorithms used for synthesis [3] . Tractability has enabled use of the GR(1) fragment for robotics applications [6, 7, 11, 13].

The issue of scalability is a major bottleneck for the widespread adoption of formal methods in robotics applications [12]. In [17], synthesis algorithms for a fragment of LTL are presented, where policy synthesis is accelerated by performing computations directly on the original system. However, unlike the GR(1) fragment, this fragment does not let us reason over disjunctions of formulas In [18], scalability is addressed using ideas from receding horizon control for synthesis from LTL. In [1], a compositional approach is taken where first a parameterized controller is synthesized and then a controller is synthesized over the parameters to compose the parametric controllers for reachability and safety specifications.

In our work, we adopt a compositional approach where the objective is to allow for the synthesis procedure to be parallelized. Our work improves the scalability of synthesis for a more general fragment than that considered in [1] and we do this by identifying sets of *equicontrollable* states. In contrast to [1], where the authors synthesize policies while reasoning about safety and reachability, the work here addresses liveness assumptions and guarantees as well. The parametric controllers for the individual reachability games are synthesized in parallel to abstract the states corresponding to the liveness guarantees into sets of equicontrollable states. Following the decomposition into such sets, we abstract away the local transitions to construct a composite synthesis problem. The performance gain comes from solving the parameterized reachability games in parallel during the identification of the equicontrollable sets.

The main contribution of this paper is an approach to decompose and parallelize synthesis for the GR(1) fragment. The approach is sound but as a limitation we do lose completeness as a result of the decomposition. The paper is organized as follows. Section 2 provides background and introduces notation and Section 3 develops an example used to illustrate the ideas presented in the paper. Section 4 describes the approach for partitioning a set into subsets of equicontrollable states. Section 5 describes a procedure for setting up the synthesis problem for the composite controller and employs the synthesized controller to compose the reachability games. This ensures that the liveness properties are satisfied (soundness). Section 6 summarizes the results from benchmarking experiments on planning problems for robots with mutually exclusive access to critical areas of the workspace.

2 Preliminaries

In this section we briefly introduce the notation that we use. Additional details and precise definitions can be found in [2, 15]. Atomic propositions are statements that evaluate to `True` or `False`. Consider a finite set of atomic propositions AP. Denote by Σ the set of states of the system ($\Sigma := 2^{\mathrm{AP}}$). We denote the restriction of the set $\mathscr{X}$ to $\mathscr{Y}$ by $\mathscr{X}|_{\mathscr{Y}}$ i.e. $\mathscr{X}|_{\mathscr{Y}} = \mathscr{X} \cap \mathscr{Y}$.

We write $s \models p$ if a state $s \in \Sigma$ satisfies a proposition $p \in \mathrm{AP}$. A state $s \in \Sigma$ satisfies a proposition $p \in \mathrm{AP}$ if and only if $p \in s$. We will work with the Boolean operators $\wedge$(conjunction), $\vee$(disjunction), $\rightarrow$(implication) and $\leftrightarrow$(bi-implication) to construct Boolean formulas. The temporal operators we use are *next* ($\bigcirc$), *eventually* ($\Diamond$) and *always* ($\Box$). For a Boolean formula ξ over AP, by $[\![\xi]\!]$ we refer to the set of states satisfying ξ. The semantics of LTL are defined over infinite words in Σ^{ω}. For $\sigma \in \Sigma^{\omega}$, σ_k refers to the $(k+1)^{\text{th}}$ element in the sequence σ with σ_0 being the first element.

For reactive synthesis, we model the synthesis problem as a two-player game where the environment satisfies certain assumptions on its behavior and with these assumptions being satisfied, the system is required to behave in a desired manner while reacting to the environment. To formulate the reactive synthesis problem, we first partition AP into two disjoint sets of variables $\mathrm{AP_e}$ and $\mathrm{AP_a}$ such that the set $\mathrm{AP_e}$ is controlled by the environment and $\mathrm{AP_a}$ is controlled by the agent being designed. The sets $\mathrm{AP_e}, \mathrm{AP_a}$ form a partition of AP, i.e., $\mathrm{AP} = \mathrm{AP_e} \cup \mathrm{AP_a}$ and $\mathrm{AP_e} \cap \mathrm{AP_a} = \emptyset$. Define the state spaces over these sets of propositions as $\Sigma_{\mathrm{e}} := 2^{\mathrm{AP_e}}$ and $\Sigma_{\mathrm{a}} := 2^{\mathrm{AP_a}}$.

The synthesis problem is to find a function $f : (\Sigma_{\mathrm{e}} \times \Sigma \times M) \rightarrow (\Sigma_{\mathrm{a}} \times M)$ such that the sequences of states generated by this strategy satisfy a given specification φ. M is a finite set of memory values with a unique initial memory value $\underline{m}$, in other words f is a finite-memory strategy. For a finite-memory strategy f, the set of infinite sequences that occur when using f are referred to as *plays*:

$$\begin{aligned}\mathrm{Plays}(f) = \{\sigma \in \Sigma^{\omega} | \exists m \in M^{\omega} \text{ such that } m_0 = \underline{m} \text{ and } \\ \forall k \geq 0.(\sigma^{\mathrm{a}}_{k+1}, m_{k+1}) = f(\sigma^{\mathrm{e}}_{k+1}, \sigma_k, m_k)\}.\end{aligned} \tag{1}$$

A strategy is *winning* for a formula φ if and only if all plays of f satisfy the formula, and it is *input enabled*, meaning that f should be defined at the initial state-memory pair, as well as at any state-memory pair that can be reached in any play. A state is *winning state* for a specification if there exists a strategy that is winning with the given state as the initial state.

2.1 Generalized Reactivity (1)

For reactive synthesis, we focus specifically on the GR(1) fragment. The GR(1) specification models a two-player game where the controlled agent has to satisfy a

set of liveness guarantees and safety constraints under some assumed behavior for the environment. This assumed behavior for the environment in turn consists of a set of liveness properties and safety constraints. A GR(1) formula has the form

$$(\theta^{\mathrm{e}} \wedge \theta^{\mathrm{a}}) \wedge (\Box\rho^{\mathrm{e}} \wedge \bigwedge_{j=1}^{m} \Box\Diamond\psi_j^{\mathrm{e}}) \rightarrow (\Box\rho^{\mathrm{a}} \wedge \bigwedge_{i=1}^{m} \Box\Diamond\psi_i^{\mathrm{a}}), \tag{2}$$

where θ^{e} is a Boolean function of propositions in AP and marks the set of assumed initial poses for the environment. θ^{a} is defined similarly for the controlled agent. ρ^{e} is the assumed safety behavior for the environment and is a Boolean function of propositions in $\mathrm{AP}_{\mathrm{a}} \cup \mathrm{AP}_{\mathrm{e}} \cup \bigcirc \mathrm{AP}_{\mathrm{e}}$, with

$$\bigcirc \mathrm{AP}_{\mathrm{e}} = \{\bigcirc\alpha : \alpha \in \mathrm{AP}_{\mathrm{e}}\}.$$

ρ^{a} is a Boolean function of $\mathrm{AP}_{\mathrm{a}} \cup \mathrm{AP}_{\mathrm{e}} \cup \bigcirc \mathrm{AP}_{\mathrm{e}} \cup \bigcirc \mathrm{AP}_{\mathrm{a}}$, while $\psi^{\mathrm{a}}, \psi^{\mathrm{e}}$ are Boolean functions of $\mathrm{AP}_{\mathrm{a}} \cup \mathrm{AP}_{\mathrm{e}}$. A GR(1) synthesis problem is to find a strategy f that is winning for this formula and, in addition, the following must hold for the every play σ of the strategy:

$$\sigma \models \boxminus\rho^{\mathrm{e}} \rightarrow \boxminus\rho^{\mathrm{a}}, \tag{3}$$

where $\boxminus$ is the *historically* temporal operator [14]. This ensures that the agent does not violate its safety constraint by forcing the environment to violate its assumption in the future.

2.2 Complexity for Synthesis

The GR(1) fragment is often used for high-level reasoning because of the polynomial-time symbolic algorithms available for the synthesis of strategies for this fragment. Symbolic algorithms allow for reasoning about problems with very large state spaces because they construct strategies by manipulating sets of states, as opposed to an enumerative approach where all the states are stored and searched. For the algorithm outlined in [3] for GR(1) synthesis, the sets are stored and manipulated as binary decision diagrams (BDDs). BDDs serve as compact representations of sets, but the variable ordering can have a significant effect on their size [2].

The complexity for reordering of BDDs is often not taken into account while analyzing the complexity of symbolic synthesis algorithms [3]. Finding the optimal variable ordering that minimizes the size of reduced order BDDs is NP-hard [4]. For a brief introduction to BDDs and their use in symbolic model checking we refer the reader to [2].

The synthesis algorithm outlined in [3] and its implementation in modern solvers [8] results in cubic time algorithms for solving the nested fixpoints. However, using ideas from [5], the nested fixpoints can be solved in quadratic time but this also results in the storing and reordering of $nm|\Sigma|^2$ BDDs (in the worst case).

Problem Statement 1. *Synthesize a strategy that is winning for a GR(1) formula φ in a scalable manner, where sub-strategies are computed in parallel and a strategy to compose them in a manner that satisfies the GR(1) formula is synthesized.*

Besides the obvious gain from parallelization, the decomposition-based approach presented here also allows us to compute, reorder, and store the BDDs for the sub-strategies separately, leading to gains in performance.

3 Example

In this section we introduce a simple instance of planning for a domestic mobile robot that we use to illustrate and develop ideas presented in the paper. Consider the workspace shown in Figure 1. The door is controlled by the environment. Rooms with charging stations have a lighting sign to indicate the same.

Door Open is the proposition that indicates whether the door is open. The robot's position is controlled by us and the robot can transition from its current position through an open slit to any of the adjacent rooms. The movement between the *Corridor* and the *Living Room* is controlled by the *Door Open* guarding the slit:

$$(\text{Door Open} \wedge \text{Corridor}) \rightarrow \bigcirc(\text{Living Room} \vee \text{Garage} \vee \text{Corridor}),$$

$$(\neg\text{Door Open} \wedge \text{Corridor}) \rightarrow \bigcirc(\text{Garage} \vee \text{Corridor}),$$

$$(\text{Door Open} \wedge \text{Living Room}) \rightarrow \bigcirc(\text{Living Room} \vee \text{Corridor}),$$

$$(\neg\text{Door Open} \wedge \text{Living Room}) \rightarrow \bigcirc(\text{Living Room}).$$

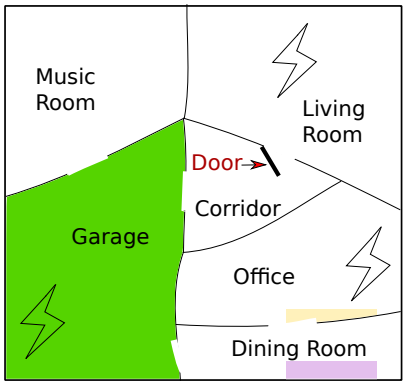


Fig. 1 Workspace for example in Section 3. Slits represent pathways for the robot. A door guards the path between the corridor and the living room. Lighting sign in a room indicates the presence of a charging station.

4 Separation into Sets of Equicontrollable States

Let φ^{e} be the assumption on the behavior of the environment and ρ^{a} be the set of transition rules for the controlled agent. Note that for a GR(1) game, φ^{e} has the form

$$\varphi^{\mathrm{e}} := \theta^{\mathrm{e}} \wedge \Box\rho^{\mathrm{e}} \wedge \bigwedge_{j=1}^{m} \Box\Diamond\psi_j^{\mathrm{e}}.$$

Definition 1. For a set of states $\mathscr{B} \subseteq \Sigma$, we denote the set of winning states for the condition in equation (4) as *WinSet*($\mathscr{B}$), or alternatively as *WinSet* of $\mathscr{B}$.

$$\varphi^{\mathrm{e}} \to \Box\rho^{\mathrm{a}} \wedge \left(\bigvee_{s\in\mathscr{B}} \Diamond s \right). \tag{4}$$

In other words, *WinSet*($\mathscr{B}$) for a set $\mathscr{B}$ is the set of states from where the agent can force the system to transition into $\mathscr{B}$ for all admissible behavior for the environment. Computing the *WinSet* for $\mathscr{B} \subseteq \Sigma$ takes at most $\mathscr{O}(m|\Sigma|^2)$ symbolic steps [10]. We will sometimes refer to synthesis problems with winning conditions similar to that in equation (4), where the objective is to force the execution to a reach a given set of states, as *reachability games*.

Definition 2. $s_1, s_2 \in \Sigma$ are *equicontrollable* if and only if $s_1 \in \mathit{WinSet}(s_2)$ and $s_2 \in \mathit{WinSet}(s_1)$.

In other words, two states are said to be equicontrollable if bidirectional reachability holds.

Definition 3. Given transition rules for the controlled agent (ρ^{a}) and the environment transition (ρ^{e}), the set of *reachable states* (Σ^{reach}) is the set of states in Σ that can be visited through any sequence of valid actions for the environment and the controlled agent.

Formally,

$$\Sigma_G^{\mathrm{reach}} = \{v | \exists X \in \Sigma_{\mathrm{e}}^*, \exists Y \in \Sigma_{\mathrm{a}}^* \text{ such that } (X_0, Y_0) \models \theta, v = (X_{-1}, Y_{-1}), |X| = |Y| \text{ and } \forall k < |X| - 1. (X_k, Y_k, X_{k+1}) \models \rho^{\mathrm{e}}, (X_k, Y_k, X_{k+1}, Y_{k+1}) \models \rho^{\mathrm{a}}\},$$

where Σ_{a}^* is the Kleene closure of Σ_{a} and $|X|$ is the length of the sequence X. The set of reachable states can be computed in at most $\mathscr{O}(|\Sigma|)$ symbolic steps. For the example in Section 3, the state $\{\mathrm{Office}, \mathrm{Garage}\}$ is not reachable because the robot can be present in either the office or the garage, but not both.

4.1 Parameterized Reachability Games

The reachability games are parameterized in a manner similar to that in [1]. However, in contrast to [1], we allow for liveness properties in addition to assuming

safety constraints for the environment. Let $\mathscr{P}_{\mathrm{AP}}$ be a set of atomic propositions introduced such that $|\mathscr{P}_{\mathrm{AP}}| = |\mathrm{AP}|$ and $\mathscr{P}_{\mathrm{AP}} \cap \mathrm{AP} = \emptyset$. Define a one-to-one and onto function $f_{\mathrm{param}} : \mathrm{AP} \to \mathscr{P}_{\Sigma}$. Consider some subset of AP over which we want to parameterize the reachability game. Denote this subset as T. For a set $T \subseteq \mathrm{AP}$, define $\mathscr{P}_T := \{t : \exists x \in T, t = f_{\mathrm{param}}(x)\}$. The augmented set of variables now is $\mathrm{AP}^{\mathscr{P}_T} = \mathscr{P}_T \cup \mathrm{AP}$. We assume the new variables introduced are controlled by the agent i.e. $\mathrm{AP}_{\mathrm{a}}^{\mathscr{P}_T} = \mathrm{AP}_{\mathrm{a}} \cup \mathscr{P}_T$ and $\mathrm{AP}_{\mathrm{e}}^{\mathscr{P}_T} = \mathrm{AP}_{\mathrm{e}}$. Define the new transition rule for the agent:

$$\bar{\rho}^{\mathrm{a}} = \rho^{\mathrm{a}} \wedge \bigwedge_{p \in \mathscr{P}_T} (p \leftrightarrow \bigcirc p).$$

The parametric propositions introduced are constrained to stay fixed during execution, in addition to the original constraints on the agent's behavior. The transition rules for the environment stay unaltered.

Consider a reachability game with the winning condition:

$$\psi_f := \varphi^{\mathrm{e}} \to \Box \rho^{\mathrm{a}} \wedge \Diamond \bigwedge_{t \in T} \left(t \leftrightarrow f_{\mathrm{param}}(t)\right) \tag{5}$$

Solving for the set of winning states for this reachability game returns the set of admissible parameters and the corresponding states in Σ that, in combination with the admitted parameters, are winning for condition (5). If $s \in \Sigma$ and and $r \in 2^{\mathscr{P}_T}$ are winning for condition (5), then what this implies is that starting from s, the controlled agent can force the execution to transition into a state satisfying $f_{\mathrm{param}}^{-1}(r)$. Note that $f_{\mathrm{param}}^{-1}(r) \subseteq T$ may only partially constrain the propositions in AP. Hence, $f_{\mathrm{param}}^{-1}(r)$ can be satisfied by multiple states in Σ.

Remark 1. The set of winning states for condition (5) can be computed in $\mathscr{O}(|\Sigma|^2)$ symbolic steps in the worst case.

This follows as a direct consequence of Lemma 9 from [10]. From the μ-calculus formula in [10], we note that the non-parameterized reachability game takes worst case $\mathscr{O}(|\Sigma|^2)$ steps. For a valuation of the parameters (r), the parameterized reachability game corresponds to solving a reachability game with $f_{\mathrm{param}}^{-1}(r)$ as the set of states to be reached. Thus, the symbolic set operations can be seen as operating on copies of the same transition system for different valuations of the parametric propositions in parallel [1]. Since the parameters stay fixed during execution, adding the parameters does not result in an increase in the number of symbolic steps needed for solving a non-parameterized reachability game. However, the symbolic steps themselves are more expensive because of the added parameters.

Example 1. Consider a system with $\mathrm{AP} = \{a, b, c\}$ and $\Sigma = 2^{\mathrm{AP}}$. $\mathscr{P}_{\mathrm{AP}} = \{p_a, p_b, p_c\}$ and for $r \in \mathrm{AP}$, f_{param} is defined as $f_{\mathrm{param}}(r) = p_r$. We seek to parameterize the *WinSet* computation over $T = \{b, c\}$, therefore we set $\mathscr{P}_T = \{p_b, p_c\}$. The state $\{a, b\}$ is in *WinSet* of the states satisfying $(b \wedge c)$ if and only if $\{b, p_b, p_c\}$ is a winning state for the condition in equation (5). This implies that with $\{b\}$ as the initial state, the agent can force the execution to a state satisfying $(b \wedge c)$.

4.2 Partitioning a Set into Equicontrollable Sets

Problem Statement 2. *Partition the set of states in Σ satisfying the Boolean formula ξ over propositions in* AP *into equicontrollable classes over the set of propositions $\mathscr{X}$.*

By partitioning over $\mathscr{X}$, we imply that for any $x_1, x_2 \subseteq \mathscr{X}$ with $x_1 \neq x_2$, the sets of states $\mathscr{S}_1 = \{s | s \in \Sigma, s|_X = x_1\}$ and $\mathscr{S}_2 = \{s | s \in \Sigma, s|_X = x_2\}$ are in the same equicontrollable class if and only if from every $s \in \mathscr{S}_1$, the agent can force the execution into $\mathscr{S}_2$ and vice versa. We slightly abuse the definition of an equicontrollable class by allowing for the states associated with $x \subseteq \mathscr{X}$ to be in the same class even though every pair of states $s_1, s_2 \in \{s | s \in \Sigma, s|_X = x\}$ may not be equicontrollable. This is done since we are interested in partitioning over $\mathscr{X}$.

In this paper, we set $\mathscr{X}$ to be the set of support propositions for the formula ξ, where $[\![\xi]\!]$ is the set of states to be separated into equicontrollable classes. For a specific application, domain knowledge could guide the selection of the set of the propositions $\mathscr{X}$ to be different from the support variables for the formula ξ.

Algorithm 1 formally describes the procedure for solving the partitioning problem. Let $\mathscr{X}$ be the propositions over which we want to separate the equicontrollable classes. First, consider the following formula:

$$\varphi^{\mathrm{e}} \to \square \rho^{\mathrm{a}} \wedge \Diamond \left(\xi \wedge \bigwedge_{t \in \mathscr{X}} \left(t \leftrightarrow f_{\mathrm{param}}(t) \right) \right). \tag{6}$$

Solving for the winning states of the above parameterized reachability game gives us a set of states of the form (s, r) with $s \in \Sigma$ and $r \subseteq \mathscr{P}_{\mathscr{X}}$. By construction, these states have the property that $f^{-1}_{\mathrm{param}}(r) \models \xi$ and from the state s, the controlled agent can force the execution to reach the set of states $\{s \in \Sigma : s|_{\mathscr{X}} = f^{-1}_{\mathrm{param}}(r)\}$.

Following this construction, we iterate through the values for the parameters (recall that the parametric propositions have a direct correspondence with the variables in $\mathscr{X}$) and split them into equicontrollable classes as in Algorithm 1. This requires us to perform at most $\mathscr{O}(k|\Sigma_{\mathscr{X}}|)$ evaluations once we have computed the winning set for (6), where k is the number of classes. Furthermore, while iterating over sets of states, we can eliminate spurious classes by ignoring those sets that have no elements in common with Σ_{reach}.

Example 2. For the workspace in Section 3, we want to partition the set of states where the robot is in a room with a charging station into equivalence classes. ξ has the form:

$$\xi = \text{Office} \vee \text{Living Room} \vee \text{Garage}.$$

This specification specifies whether the room the robot is in has a charging station. And we set

$$\mathscr{X} = \{\text{Office, Living Room, Garage}\},$$

the supporting propositions for ξ.

Suppose $\varphi^{\mathrm{e}} = \texttt{True}$ i.e the behavior of the door is unconstrained. This yields that Garage, Office are in the same equicontrollable class while Living Room is in a different class. When we assume that the door opens infinitely often ($\varphi^{\mathrm{e}} = \square\diamond$Door Open) as a model for the environments behavior, the states corresponding to Garage, Office and Living Room are in the same equicontrollable class.

Algorithm 1 Separating into equicontrollable Classes

Input: • Environmental Behavior φ^{e}, System safety/transition rules ρ^{a}.
- Specification ξ representing the set of states to be separated ($[\![\xi]\!]$).
- BDD ρ^{reach} representing the set of reachable states for the system.
- Set of propositions $\mathscr{X} \subseteq$ AP over which the states must be partitioned and the map f_{param}.

Output: • Equicontrollable classes $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_k$ s.t. $\alpha_i \cap \alpha_j = \emptyset$ for $i \neq j$, $\bigcup_{l=1}^{k} \alpha_i = [\![\xi]\!]$.

Define $\varphi_{\xi}^{\mathrm{param}} := \varphi^{\mathrm{e}} \rightarrow \square\rho^{\mathrm{a}} \wedge \diamond \left(\xi \wedge \bigwedge_{t \in \mathscr{X}} \left(t \leftrightarrow f_{\mathrm{param}}(t) \right) \right)$
Compute winning states ($W_{\varphi_{\xi}^{\mathrm{param}}}$) for $\varphi_{\xi}^{\mathrm{param}}$
Equicontrollable Classes $= \emptyset$
for $x \subseteq \mathscr{X}$ **do**
  $t_1 = f_{\mathrm{param}}^{-1}(x)$; $EquivFlag = 0$
  for $p \in$ Equicontrollable Classes **do**
    $t_2 = f_{\mathrm{param}}^{-1}(p)$
    if $\left(\exists s.s|_{\mathscr{X}} = x \wedge (s,t_2) \in W_{\varphi_{\xi}^{\mathrm{param}}} \wedge \exists s.s|_{\mathscr{X}} = p \wedge (p,t_1) \in W_{\varphi_{\xi}^{\mathrm{param}}} \right)$ **then**
      $EquivFlag = 1$
    end if
  end for
  if $EquivFlag = 0$ and $(\exists s \in \Sigma.s \models \rho^{\mathrm{reach}} \wedge s|_{\mathscr{X}} = x)$ **then**
    Equicontrollable Classes = Equicontrollable Classes $\cup \{s | s \in \Sigma, s|_{\mathscr{X}} = x\}$
  end if
end for
return Equicontrollable Classes

5 Compositional Synthesis

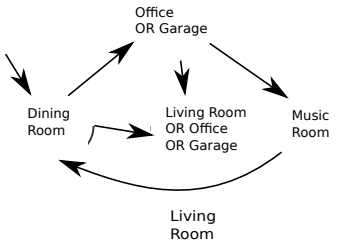
This section describes an approach to build a transition system and a specification such that the winning strategy for this system can be used to compose the sub-strategies that are synthesized and stored in parallel to find a strategy winning against a GR(1) specification.

Example 3. For the workspace from Section 3, consider a synthesis problem where the robot has to patrol the dining room and the music room infinitely often, while making sure to visit a room with a charging station infinitely often and the the robot is initially in the dining room. The liveness guarantees to be satisfied are:

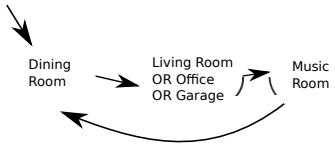
$$\square\diamond \text{ Dining Room}\,, \quad \square\diamond (\text{Office} \vee \text{Garage} \vee \text{Living Room})\,, \quad \square\diamond \text{Music Room}.$$

We fix the ordering of liveness guarantees (Dining Room, Office $\vee$ Garage $\vee$ Living Room, Music Room). and build a new transition system as shown in Figure 2. For each liveness guarantee, there is state in the transition system corresponding to a subset of equicontrollable classes arising from decomposition of the states satisfying the liveness guarantee. We add transitions between these states if the predecessor is in the *WinSet* of the successor.

If the environments behavior is modeled as $\varphi^{\mathrm{e}} = \texttt{True}$, we get the abstracted supervisory transition system shown in Figure 2a. For the liveness guarantee Office $\vee$ Garage $\vee$ Living Room, the classes are $\{\text{Office} \vee \text{Garage}, \text{Living Room}\}$. The transition system in Figure 2a has states corresponding to all non-empty subsets of these classes. If we assume that the door infinitely often opens, the supervisory transition system is that shown in Figure 2b. Both transition systems have a cycle that can be used to compose the sub-strategies.



(a) Door need not open infinitely often



(b) Door opens infinitely often

Fig. 2: Supervisory transition system for different environment behavior

The construction of the transition system and the composing of the sub-strategies is formally described below.

5.1 Synthesizing a Composite Controller

Consider the GR(1) formula in equation (2). The states corresponding to the liveness guarantees for the agent (ψ_i^{a}) are partitioned into equicontrollable classes as described in Section 4. For each $i \in \{1,2,\ldots,n\}$, let k_i be the number of classes ψ_i^{a} be partitioned into. Let $\Lambda_i = \{\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,k_i}\}$ be the set of classes associated with ψ_i^{a}. Define Ω_i to be set of all subsets of Λ_i except the empty set ($\emptyset$). Note that $|\Omega_i| = 2^{k_i} - 1$. Without loss of generality, we fix some ordering of the elements in Ω_i such that $\Omega_i = \{\Omega_{i,1}, \Omega_{i,2}, \ldots, \Omega_{i,2^{k_j}-1}\}$. Denote by $A_{\Omega_{i,j}}$ the *WinSet* for $\Omega_{i,j}$ i.e $A_{\Omega_{i,j}} = \mathit{WinSet}(\Omega_{i,j})$ For the case in Example 3 where the door is not assumed to open infinitely often, the classes corresponding the specification where the robot has to infinitely often visit a room with a charging station are

$$\Lambda_2 = \{\text{Office} \vee \text{Garage},\ \text{Living Room}\}$$

and

$$\Omega_2 = \Big\{ \{\text{Office} \vee \text{Garage}\}, \{\text{ Living Room}\}, \{\text{Office} \vee \text{Garage, Living Room}\} \Big\}.$$

Note that if $|\Omega_{i,j}| = 1$ we can use the parametric *WinSet* computed for decomposing $[\![\psi_i^{\text{a}}]\!]$ into equicontrollable classes to obtain the *WinSet* for $\Omega_{i,j}$ by setting values to the parametric propositions appropriately. For $\Omega_{i,j}$ with $|\Omega_{i,j}| > 1$, we compute $\mathit{WinSet}(A_{\Omega_{i,j}})$ by solving a reachability game with $\Omega_{i,j}$ as the goal to be reached. As these computations are independent, they can be performed in parallel. The symbolic steps here are less difficult than that for finding the *WinSet* set of $[\![\psi_i^{\text{a}}]\!]$, because we are only considering a subset of the set of states satisfying the liveness constraint $[\![\psi_i^{\text{a}}]\!]$ and hence there are fewer transitions to be accounted for in each symbolic step.

Following this setup, the hierarchical game is constructed as follows. The set of atomic propositions are:

$$\overline{\text{AP}} = \{\rho_{i,j_i} : i \in \{1,2,\ldots,n\}, j_i \in \{1,2,\ldots,2^{k_i}-1\}\}.$$

The transition rule is specified as:

$$\rho^{\text{compositional}} = \bigvee_{v \in \overline{\text{AP}}} v \quad \wedge \underbrace{\bigwedge}_{\substack{i \in \{1,2,\ldots,n\} \\ j \in \{1,2,\ldots,2^{k_i}-1\}}} \left(\rho_{i,j} \rightarrow \bigcirc \underbrace{\bigvee}_{\substack{\Omega_{i,j} \subseteq A_{\Omega_{k,l}} \\ k \in \{i, i \oplus 1\} \\ l \in \{1,2,\ldots,2^{k_i}-1\}}} \rho_{k,l} \right). \tag{7}$$

Here $\bigvee$ is the *XOR* operator. In equation (7), we allow for a transitions between the states $\{\rho_{i,j}\}$ and $\{\rho_{k,l}\}$ only if the current $\Omega_{i,j}$ is in the *WinSet* for the $\Omega_{k,l}$ corresponding to the successor state. For the transition system in Figure 2a, the Dining Room is in the *WinSet* of Living Room $\vee$ Office $\vee$ Garage but Living Room $\vee$ Office $\vee$ Garage is not in the *WinSet* of Music Room. The transition relations reflect the same. Similarly, transition relations are constructed between the other states.

Note that we restrict $k \in \{i, i \oplus 1\}$, ensuring that only those transitions are chosen that either stay in the same liveness guarantee or lead to the next liveness guarantee. This way we do not cycle back to a liveness guarantee that was visited earlier in the current cycle. This makes the transition rules sparse, keeping the BDD small, reducing the time required for synthesizing the composite controller.

The liveness guarantees ensure that infinitely often for each $i \in \{1,2,\ldots,n\}$, $\rho_{i,j}$ for some j is satisfied. The liveness guarantees can be formally written as:

$$\psi_i^{\text{compositional}} := \bigvee_{j \in \{1,2,\ldots,2^{k_i}-1\}} \rho_{i,j}. \tag{8}$$

This ensures that at least one of the classes corresponding to a liveness guarantee is visited in each cycle through the liveness guarantees. For a particular i, satisfying $\psi_i^{\text{compositional}}$ is equivalent to satisfying ψ_i^{a} in the original system.

Note that here there is no environment here and we only need to search for a cycle passing through all the liveness guarantees for a given set of initial states. Consider some $i \in \{1,2,\ldots,n\}$. The set of valid initial states are the nodes corresponding to the elements in Ω_i for which $[\![\theta^{\text{a}} \wedge \theta^{\text{e}}]\!]$ lies in their *WinSet*. The initial condition can be written as (for some i):

$$\theta^{\text{compositional}} := \bigvee_{\theta \in A_{\Omega_{i,j}}} \rho_{i,j} \,. \tag{9}$$

Composing the specifications above, we need to find a controller for the condition:

$$\theta^{\text{compositional}} \wedge \Box \rho^{\text{compositional}} \wedge \bigwedge_{i=1}^{n} \Box\Diamond \psi_i^{\text{compositional}} \,. \tag{10}$$

Finding a winning strategy $f^{\text{compositional}} : \overline{\text{AP}} \times M^{\text{sup}} \to \overline{\text{AP}} \times M^{\text{sup}}$ for the above specification gives the compositional controller for composing the strategies for the reachability games.

5.2 Composing the Sub-strategies

Define $f_{i,l}^{k,l} : \Sigma_{\text{e}} \times \Sigma \times M_{i,l}^{k,l} \to \Sigma_{\text{a}} \times M_{i,l}^{k,l}$ to be the strategy that takes the agent from a state in $\Omega_{i,l}$ to $\Omega_{k,l}$. Let $\underline{m}_{i,j}^{k,l}$ be the initial memory value for $M_{i,l}^{k,l}$. Without loss of generality, assume $M_{i_1,l_1}^{k_1,l_1} \cap M_{i_2,l_2}^{k_2,l_2} = \emptyset$ when $(i_1,j_1,k_1,l_1) \neq (i_2,j_2,k_2,l_2)$.

Define $k_{\max} := \max\{k_i : i \in \{1,2,\ldots,n\}\}$, i.e. $k_{\max}$ is the size of the largest number of equicontrollable classes for any of the liveness classes. Let $\overline{M} := M^{\text{sup}} \times \bigcap_{i,j,k,l} M_{i,j}^{k,l}$ and $\xi_{\text{comp}} := \{1,2,\ldots,N\} \times \{1,2,\ldots,k_{\max}\} \times \{1,\ldots,N\} \times \{1,\ldots,k_{\max}\}$.

We construct a strategy

$$f^{\text{compose}} : \Sigma_{\text{e}} \times \Sigma \times \xi_{\text{comp}} \times \overline{M} \to \Sigma_{\text{a}} \times \xi_{\text{comp}} \times \overline{M}$$

that uses $f^{\text{compositional}}$ to compose the strategies for the reachability games ($f_{i,l}^{k,l}$):

$$f^{\text{compose}}(x,s,i,j,k,l,w,w_{\text{sup}}) = (y,i',j',k',l',w',w'_{\text{sup}}), \tag{11}$$

where if $s \notin \Omega_{k,l}$, then

$$\begin{aligned} (y,w') &= f_{i,j}^{k,l}(x,s,w), \\ (i',j',k',l') &= (i,j,k,l), \\ w'_{\text{sup}} &= w_{\text{sup}}, \end{aligned} \tag{12}$$

and if $s \in \Omega_{k,l}$, then

$$\begin{aligned} (y, w') &= f_{i,j}^{k,l}(x, s, \underline{m}_{i,j}^{k,l}), \\ (\{\rho_{k',l'}\}, w'_{\text{sup}}) &= f^{\text{compositional}}(\{\rho_{k,l}\}, w_{\text{sup}}), \\ (i', j') &= (k, l). \end{aligned} \tag{13}$$

In equation (12), while we are moving towards $\Omega_{k,l}$, the values are updated according to the strategy $f_{l,j}^{k,l}$. Once we reach $\Omega_{k,l}$ (equation (13)), the next goal is updated according to $f^{\text{compositional}}$ and we continue towards the next goal, switching goals again once the next goal is reached.

Theorem 1. *Strategy f^{compose} is sound. Solving equation* (10) *takes in the worst case $\mathcal{O}((2^{k_{max}})^2 n^3)$ symbolic steps.*

Proof. By construction, a winning strategy in the original system was computed corresponding to every transition in the abstracted system i.e the agent can either force the execution to the next liveness guarantee or block the environment from satisfying the assumption on its behavior. A winning strategy for the abstracted system finds an execution that cycles through the liveness guarantees. Cycling through the liveness guarantees in the abstracted system correspond to cycling through liveness guarantees in the original system. Hence, composing the strategies from the reachability games in accordance with the composite controller ensures satisfaction of the original GR(1) formula.

The specification resulting in equation (10) is a GR(1) formula without an environment i.e it is not reactive, hence the innermost fixpoint associated with blocking the environment from satisfying its assumptions does not add to the number of symbolic steps to be performed. The total number of states is $(n2^{k_{max}})$ and there are n liveness guarantees, resulting in $\mathcal{O}((2^{k_{max}})^2 n^3)$ symbolic steps [9]. □

For applications where the number of liveness guarantees and the number of equicontrollable classes are much smaller than the total number of states, i.e $n \ll |\Sigma|$ and $k \ll |\Sigma|$, the parallelized approach presented here is well-suited and should result in performance gains in term of computation time. We expect such behavior in multi-agent systems with large state spaces where the agents' dynamics are not closely coupled.

Limitations

There can be potential corner cases where the algorithm presented above is not complete[1] as we lose certain transitions during abstraction into the supervisory transition system. Besides completeness another limitation of the approach is that if we end up with a large number of equicontrollable classes, the computation of the compositional strategy can become intractable.

[1] See `https://dathath.github.io/papers/appendix.pdf` for example demonstrating incompleteness of the approach.

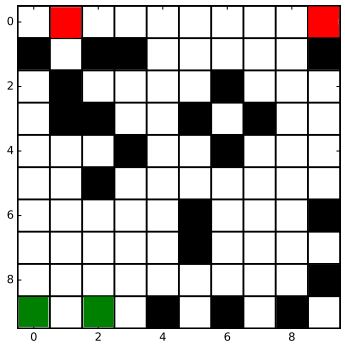
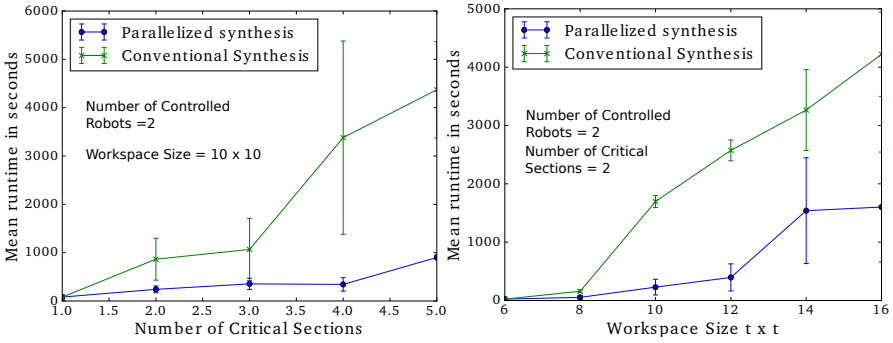


Fig. 3 Workspace with shaded obstacles (black) and critical sections (green and red).

6 Experiments

We consider a multi-agent robot motion planning problem where the objective is to, for a set of robots, schedule access to critical sections of a given workspace in a safe manner. The environment consists of an uncontrolled adversarial mobile robot that is functioning in the same workspace as the controlled robots and requires access to certain critical sections of the workspace. An example instance is shown in Figure 3. Both the controlled robots and the uncontrolled robot have to visit cells shaded with each of the three colors (red and green) infinitely often. The problem instances are parameterized in terms of the size of the workspace, the number of critical sections and the number of controlled robots. The colored cells represent critical sections of the workspace that must be accessed in a mutually exclusive manner and each color represents a critical resource of a type. While the adversarial robot is accessing a critical section, the controlled robots must not access the same critical section. Similarly, no two controlled robots can access a critical section at the same time. The adversarial robot's access to the critical sections is prioritized over the controlled robots. When the adversarial robot attempts to access a critical space, the controlled robots as a part of their safety requirement must allow the adversarial robot to gain access by vacating the critical section. The regions shaded black (density=5%) represent static obstacles and both the uncontrolled and controlled robots must avoid the obstacles. The robots are allowed to transition to any of their non-diagonally adjacent cells in a single step. The uncontrolled robot is allowed to pursue a trajectory of its choice and the only assumption on its behavior, in addition to the constraints on its motion, is that the uncontrolled robot will access cells shaded with each of the colors infinitely often.

Figures 4a, 4b report performance over problem instances of varying size. The mean time over 50 problem instances is reported. For each of these instances the initial positions of the robots, the positions of the obstacles and the critical sections are randomized. The computations were performed on a 32 core AMD Opteron machine at 2.4GHz with 96 GB of RAM, and and we see considerable gains in computation time for the parallelized approach.



(a) Varying Number of Critical Sections

(b) Varying size of the workspace

Fig. 4: Performance on benchmark experiments – mean runtimes over 50 randomized problem instances are reported, error bars indicate standard deviation.

7 Conclusion and Future Work

A major challenge to the widespread adoption of formal methods is their scalability. As systems get larger and complex, scalable algorithms that can deal with the size and complexity of the systems are necessary. In this regard, we present an approach that allows us to decompose and parallelize the synthesis algorithm for the GR(1) fragment of linear temporal logic. The approach relies on the construction of a composite strategy that is used to compose local strategies to ensure satisfaction of the GR(1) specification. However, the approach comes with certain drawbacks as outlined earlier. Empirical evidence demonstrating the resulting gains in performance is presented for a robot motion planning problem, where in addition to path planning, safe access to certain critical sections of the workspace is scheduled.

Future work would be to explore if a similar approach can be used to synthesize policies that can handle uncertainty, because some local uncertainty can be tolerated without a resynthesis of the entire strategy by applying a local correction. Some unexpected disturbances can be handled locally, without the need for global synthesis. A hierarchical framework as presented here could be used in such settings. Another direction for future work includes exploring the possibility of a symbolic approach for decomposition of sets into equicontrollable classes as opposed to the enumerative approach considered here.

Acknowledgements This work was supported by STARnet, a Semiconductor Research Corporation program, sponsored by MARCO and DARPA.

References

1. R. Alur, S. Moarref, and U. Topcu. Compositional synthesis with parametric reactive controllers. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, HSCC 2016, Vienna, Austria, April 12-14, 2016*, pages 215–224, 2016.
2. C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.
3. R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. *Journal of Computer and System Sciences*, 78:911–938, May 2012.
4. B. Bollig and I. Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on Computers*, 45(9):993–1002, Sep 1996.
5. A. Browne, E. Clarke, S. Jha, D. Long, and W. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theoretical Computer Science*, 178(1):237 – 255, 1997.
6. S. Chinchali, S. C. Livingston, U. Topcu, J. W. Burdick, and R. M. Murray. Towards formal synthesis of reactive controllers for dexterous robotic manipulation. In *2012 IEEE International Conference on Robotics and Automation*, pages 5183–5189, May 2012.
7. J. A. DeCastro, J. Alonso-Mora, V. Raman, D. Rus, and H. Kress-Gazit. Collision-free reactive mission and motion planning for multi-robot systems. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, Sestri Levante, Italy, September 2015.
8. R. Ehlers and V. Raman. Slugs: Extensible GR(1) Synthesis. In S. Chaudhuri and A. Farzan, editors, *Computer Aided Verification: 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II*, pages 333–339, Cham, 2016. Springer International Publishing.
9. E. A. Emerson and C. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Proc. Symp. on Logic in Computer Science*, Cambridge, MA, 1986.
10. Y. Kesten, N. Piterman, and A. Pnueli. Bridging the gap between fair simulation and trace inclusion. *Information and Computation*, 200:35–61, 2005.
11. H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Where's Waldo? Sensor-Based Temporal Logic Motion Planning. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3116–3121, April 2007.
12. R. Majumdar. Robots at the edge of the cloud. In *Proceedings of the 22Nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems - Volume 9636*, pages 3–13, New York, NY, USA, 2016. Springer-Verlag New York, Inc.
13. S. Maniatopoulos, P. Schillinger, V. Pong, D. C. Conner, and H. Kress-Gazit. Reactive high-level behavior synthesis for an atlas humanoid robot. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4192–4199, May 2016.
14. Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *(PODC '90) Proceedings of the ninth annual ACM Symposium on Principles of Distributed Computing*, pages 377–408, 1990.
15. A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.
16. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '89, pages 179–190, New York, NY, USA, 1989. ACM.
17. E. M. Wolff and R. M. Murray. Optimal control of nonlinear systems with temporal logic specifications. In M. Inaba and P. Corke, editors, *Robotics Research: The 16th International Symposium ISRR*, pages 21–37. Springer International Publishing, 2016.
18. T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding Horizon Temporal Logic Planning. *IEEE Transactions on Automatic Control*, 57(11):2817–2830, Nov 2012.